Modeling, analyzing and controlling hybrid systems by Guarded Flexible Nets

Jorge Júlvez[a,b,c,*], Stephen G Oliver[a,b]

[a] *Cambridge Systems Biology Centre, University of Cambridge, Cambridge, UK*
[b] *Department of Biochemistry, University of Cambridge, Cambridge, UK*
[c] *Department of Computer Science and Systems Engineering, University of Zaragoza, Zaragoza, Spain*

Abstract

A number of artificial and natural systems can be modeled as hybrid models in which continuous and discrete variables interact. Such hybrid models are usually challenging to analyze and control due to the computational complexity associated with existing methods. In this paper, the novel modeling formalism of *Guarded Flexible Nets* (GFNs) is proposed for the modeling, analysis and control of hybrid system. A GFN consists of an *event net* that determines how the state changes as processes execute, and an *intensity net* that determines the speeds of the processes. In a GFN, the continuous state is given by the value of its state variables, and the discrete state is given by the region within which such variables lie. GFNs are shown to possess a high modeling power while offering appealing analysis and control possibilities.

Keywords: Guarded Flexible Nets, Piecewise Linear Systems, Petri Nets, Uncertain Parameters, Nondeterminism

1. Introduction

Hybrid systems are a rather general class of dynamical systems that combine continuous dynamics and discrete events [1, 2]. This generality allows for the modeling of a number of system properties. Thus, not surprisingly, hybrid systems have been used successfully to model systems in different application domains such as manufacturing, the automotive industry, computer networks, biological systems, etc. [3, 4, 5, 6, 7]. The state of a hybrid system is usually given by two sets of variables: a set of real variables accounting for the continuous state of the system, and a set of integer variables accounting

*Corresponding author. E-mail address: julvez@unizar.es

for the discrete state. The way continuous and discrete state variables interact over time depends on the adopted modeling framework (or formalism).

Among the most popular frameworks to model hybrid systems are hybrid automata [8, 9], mixed logical dynamical (MLD) systems [10], and hybrid Petri nets [11, 12]. Hybrid automata consist of a finite state machine and a set of differential equations associated with each location of the state machine. These automata have proved to be very successful for the verification and analysis of hybrid systems through model checking techniques [13]. MLD systems are computationally oriented models that describe the discrete-time evolution of the system by means of linear inequalities that include both real and binary variables. MLD systems have been shown to be equivalent to piecewise linear affine systems [14] and are very well suited for use in model predictive control [15]. Hybrid Petri nets are an extension of classical discrete Petri nets [16, 17] in which the firing of some transitions is relaxed to the real numbers. Some of the advantages of hybrid Petri nets are their ability to represent the system graphically, and their potential to tailor (or straightforwardly use) the existing toolbox of structural analysis techniques of classical Petri nets.

Despite the success of the current approaches to hybrid systems, the modeling, analysis, and control of some systems remains challenging. Among these challenges are the difficulty these approaches have in accommodating parameter uncertainties in the model, and the computational complexity associated with many analysis techniques and control tasks, i.e. the computation of control actions that achieve a given goal. This paper deals with Guarded Flexible Nets (GFNs) [18], a modelling formalism inspired by Petri nets, that attempts to alleviate these problems.

GFNs model the relationships between the state and the processes of a system by means of two nets, the event net and the intensity net: the event net specifies how the state changes when the processes of the system execute, and the intensity net specifies the speed of the processes as a function of the state of the system. Both the event net and the intensity net are tripartite graphs composed of places, transitions and handlers. These nets can accommodate uncertain parameters through sets of linear inequalities that are associated with their handlers. The combination of an event net and an intensity net results in a Flexible Net (FN) [18].

In an FN, the speeds of the processes depend linearly on the state of the system. In order to model non-linear speed functions, FNs can be enriched with guards that are associated with the intensity net, this leads to GFNs. In a GFN, the speeds of the processes depend on the guards that are active. Thus, a GFN can be seen as a hybrid system in which the continuous state is given by the marking, and the discrete state is given by the set of active

guards. As it is shown in the following sections, this simple relationship between continuous and discrete states can be exploited to model a wide variety of dynamical behaviours while keeping a relatively compact graphical notation.

In contrast to hybrid Petri nets [19, 20, 21] that allow discontinuities in the marking evolution, the marking of all the places of a GFN follows a continuous trajectory, and the discrete state is given by the set of active guards. Thus, the marking discontinuities of hybrid Petri nets cannot be mimicked by the marking of the places of a GFN. Nevertheless, particular classes of hybrid Petri nets in which the evolution of continuous places does not contain discontinuities, can be modeled by GFNs that map regions to the discrete markings of the hybrid Petri net. On the other hand, it should be noted that, in contrast to hybrid Petri nets, GFNs offer the possibility to model separately the marking changes produced by the execution of transitions, and the transitions speeds produced by the marking, through the event and intensity nets. Moreover, GFNs can handle a number of uncertain parameters, e.g. uncertain initial marking, uncertain default intensities, and uncertain dynamics modeled by the inequalities associated with both the event and the intensity handlers.

All the potential trajectories of a GFN can be accounted for by a set of necessary reachability conditions [18]. These conditions are expressed in terms of linear and quadratic inequalities that contain real and binary variables, and that relate the initial, average, and final state of the net for a given time period. In order to analyze the system, these constraints can be combined with appropriate objective functions. The solution of the resulting programming problems can be used to estimate the state of the system or to compute bounds of interest.

Control actions can be introduced straightforwardly in a GFN by means of intensity, or speed, variables associated with the transitions. The effect of these control actions on the system dynamics is established by the intensity net. A given control goal, expressed in terms of an objective function, together with the reachability conditions of the GFN, make up a programming problem whose solution contains the values of the control actions to be implemented in the system.

The paper is organized as follows: FNs and GFNs are introduced in Sections 2 and 3 respectively. Section 4 shows how hybrid systems can be modeled an analyzed by GFNs. Section 5 focuses on the control possibilities of hybrid systems by GFNs. The main conclusions are drawn in Section 6.

2. Flexible Nets

A Flexible Net (FN) is composed of an intensity net and an event net. This section first introduces intensity nets, then event nets, and finally FNs.

2.1. Intensity Nets

The intensity net determines the intensity, or speed, of the transitions as a function of the marking. Intensity nets contain three types of vertices: places, transitions and intensity handlers. Places and transitions are connected through intensity handlers which establish the relation between marking and intensity. An intensity net can be denoted as $P/S/T$ nets, which can be interpreted as: tokens in places P produce and consume intensities in transitions T through intensity handlers S. More formally:

Definition 1 (Intensity net). *An intensity net is a tuple* $\mathcal{N}_S = (P, T, S, E_S, C, D)$ *where* (P, T, S, E_S) *is a tripartite graph determining the net structure and* (C, D) *are matrices determining the potential intensity changes produced by the marking.*

The set of vertices of the net is partitioned into three sets:

- $P = \{p_1, \ldots, p_i, \ldots\}$ is a set of $|P|$ places.
- $T = \{t_1, \ldots, t_j, \ldots\}$ is a set of $|T|$ transitions.
- $S = \{s_1, \ldots, s_l, \ldots\}$ is a set of $|S|$ intensity handlers.

Similarly to Petri nets, the places model the type of components of the system and are depicted as circles, and the transitions model the processes of the system and are depicted as rectangles. The intensity handlers are depicted as dots and model the different ways in which the marking of places can generate intensities in the transitions.

The vertices of the net are connected by the edges in E_S. Each pair of vertices can be connected by at most one edge. The set E_S is partitioned into two sets E_S^T and E_S^P, where E_S^T is a set of directed edges, or simply *arcs*, connecting transitions to intensity handlers and vice versa, and E_S^P is a set of undirected edges, or simply *edges*, connecting places and intensity handlers. More formally:

- Every $e \in E_S^T$ is either an arc $e = (t_j, s_l)$ from a transition t_j to a handler s_l, or an arc $e = (s_l, t_j)$ from a handler s_l to a transition t_j.
- Every $e \in E_S^P$ is an edge $e = \{p_i, s_l\}$ connecting a place p_i and a handler s_l.

Direct connections among places and transitions are not allowed. The following notation will be used:

- ${}^{t}s_l$ denotes the input transitions of s_l, i.e. ${}^{t}s_l = \{t_j | (t_j, s_l) \in E_S^T\}$
- s_l^t denotes the output transitions of s_l, i.e. $s_l^t = \{t_j | (s_l, t_j) \in E_S^T\}$
- ${}^{s}t_j$ denotes the input handlers of t_j, i.e. ${}^{s}t_j = \{s_l | (s_l, t_j) \in E_S^T\}$
- t_j^s denotes the output handlers of t_j, i.e. $t_j^s = \{s_l | (t_j, s_l) \in E_S^T\}$
- ${}^{p}s_l$ denotes the places connected to s_l, i.e. ${}^{p}s_l = \{p_i | \{p_i, s_l\} \in E_S^P\}$
- p_i^s denotes the handlers connected to p_i, i.e. $p_i^s = \{s_l | \{p_i, s_l\} \in E_S^P\}$

Example 1. *Fig. 1(a) depicts an intensity net with* 4 *places,* p_1*,* p_2*,* p_3 *and* p_4*;* 4 *transitions,* t_1*,* t_2*,* t_3 *and* t_4*; and* 3 *intensity handlers* s_1*,* s_2 *and* s_3*. Places(transitions) are connected to intensity handlers by edges(arcs). As an example of the introduced notation, the output transitions of* s_2 *are* $s_2^t = \{t_3\}$*, and the intensity handlers connected to* p_3 *are* $p_3^s = \{s_2, s_3\}$*.*

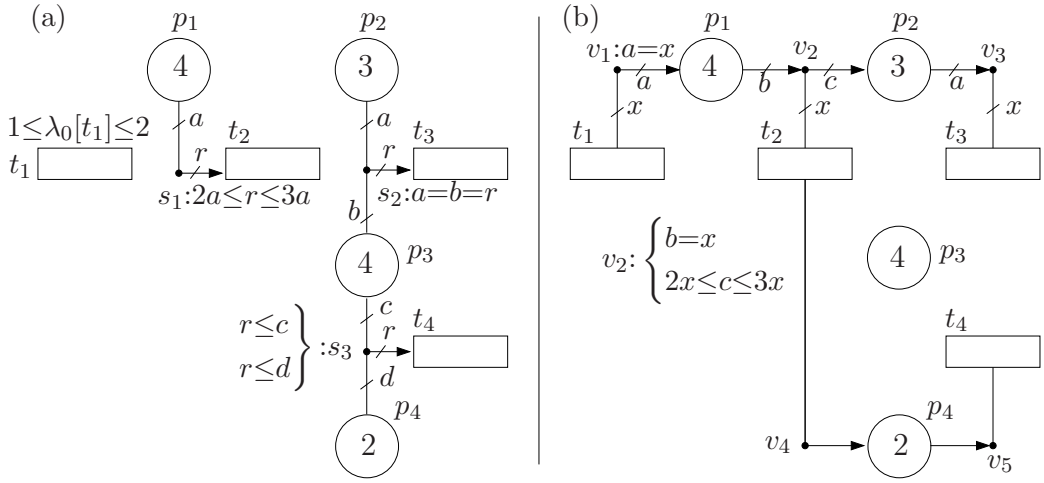

Figure 1: (a) Intensity net. (b) Event net.

Each place has a nonnegative real number of tokens (or marking) that can be used by the intensity handlers that are connected to it in order to produce intensities. A token is *active* if it is being used by an intensity handler, otherwise it is *idle*. While idle tokens are associated with places, active tokens are associated with edges. An intensity handler determines how much intensity is produced in its arcs as a function of the number of active

tokens in its edges. The intensity of a transition is obtained as function of the intensity in its arcs. In order to account for these relations, the following state variables are used:

Definition 2 (State). *The state of an intensity net $\mathcal{N}_S$ is given by the tuple $(m, \mu_P, \mu_E, \Delta\lambda, \lambda)$, where:*

- $m \in \mathbb{R}_{\geq 0}^{|P|}$ *is the marking, i.e. a vector indexed by P where $m[p_i]$ is the number of tokens in p_i,*
- $\mu_P \in \mathbb{R}_{\geq 0}^{|P|}$ *is a vector indexed by P where $\mu_P[p_i]$ is the number of idle tokens in p_i,*
- $\mu_E \in \mathbb{R}_{\geq 0}^{|E_S^P|}$ *is a vector indexed by E_S^P where $\mu_E[\{p_i, s_l\}]$ is the number of active tokens of p_i being used by s_l,*
- $\Delta\lambda \in \mathbb{R}_{\geq 0}^{|E_S^T|}$ *is a vector indexed by E_S^T where $\Delta\lambda[e]$ is the intensity in arc e. If $e = (t_j, s_l)$ then $\Delta\lambda[e]$ is a decrease of intensity in t_j produced by s_l; if $e = (s_l, t_j)$ then $\Delta\lambda[e]$ is an increase of intensity in t_j produced by s_l,*
- $\lambda \in \mathbb{R}_{\geq 0}^{|T|}$ *is a vector indexed by T where $\lambda[t_j]$ is the intensity of t_j.*

The number of tokens in a place, $m[p_i]$, is equal to the number of idle tokens in the place, $\mu_P[p_i]$, plus the number of active tokens, $\mu_E[\{p_i, s_l\}]$, in the connected edges:

$$m[p_i] = \mu_P[p_i] + \sum_{s_l \in p_i^s} \mu_E[\{p_i, s_l\}] \quad \forall\ p_i \in P \tag{1}$$

which can be expressed in matrix form as:

$$m = \mu_P + Y_m \mu_E \tag{2}$$

where Y_m is a matrix with rows indexed by P and columns indexed by E_S^P.

Example 2. *Equation (2) of the intensity net in Fig. 1(a) is:*

$$\begin{pmatrix} m[p_1] \\ m[p_2] \\ m[p_3] \\ m[p_4] \end{pmatrix} = \begin{pmatrix} \mu_P[p_1] \\ \mu_P[p_2] \\ \mu_P[p_3] \\ \mu_P[p_4] \end{pmatrix} + \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \mu_E[\{p_1, s_1\}] \\ \mu_E[\{p_2, s_2\}] \\ \mu_E[\{p_3, s_2\}] \\ \mu_E[\{p_3, s_3\}] \\ \mu_E[\{p_4, s_3\}] \end{pmatrix}$$

The relation between the number of active tokens, μ_E, and the intensities produced in arcs, $\Delta\lambda$, is given by a set of inequalities associated with each intensity handler $s_l \in S$. The coefficients of these inequalities can be captured by two matrices C and D of real numbers and same number of rows. The columns of C and D are indexed by E_S^T and E_S^P, respectively. The relation between active tokens, μ_E, and intensities in arcs, $\Delta\lambda$, can be expressed by:

$$C\Delta\lambda \leq D\mu_E \tag{3}$$

Example 3. *Consider the net in Fig. 1(a). In order to simplify the writing of inequalities, edges and arcs are associated with labels that denote number of active tokens and produced intensity. For instance, the label of edge* $\{p_1, s_1\}$ *is 'a' and denotes* $\mu_E[\{p_1, s_1\}]$*. In this way, the inequality* $2a \leq r \leq 3a$ *associated with* s_1 *means that the intensity produced in* (s_1, t_2) *is between two and three times the number of active tokens in* $\{p_1, s_1\}$ *(any value in this interval is valid). The equations associated with* s_2 *imply that active tokens both in* $\{p_2, s_2\}$ *and* $\{p_3, s_2\}$ *are required simultaneously to produce intensity in* (s_2, t_3) *(more precisely, the number of active tokens in both edges and the intensity produced are forced to be the same). The inequalities of* s_3 *mean that the intensity produced in* (s_3, t_4) *is less than or equal to* $\mu_E[\{p_3, s_3\}]$ *and* $\mu_E[\{p_4, s_3\}$*. Notice that the tokens in* p_3 *can synchronize either with tokens in* p_2 *and produce intensity in* (s_2, t_3)*, or with tokens in* p_4 *and produce intensity in* (s_3, t_4)*. Such a choice is nondeterministic and is allowed to change over time. Equation* (3) *of the net in Fig. 1(a) is:*

$$\begin{pmatrix} -1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \Delta\lambda[(s_1, t_2)] \\ \Delta\lambda[(s_2, t_3)] \\ \Delta\lambda[(s_3, t_4)] \end{pmatrix} \leq \begin{pmatrix} -2 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \mu_E[\{p_1, s_1\}] \\ \mu_E[\{p_2, s_2\}] \\ \mu_E[\{p_3, s_2\}] \\ \mu_E[\{p_3, s_3\}] \\ \mu_E[\{p_4, s_3\}] \end{pmatrix}$$

Each transition t_j is assigned a default intensity $\lambda_0[t_j]$. The intensity $\lambda[t_j]$ in a transition t_j is equal to $\lambda_0[t_j]$ plus the positive changes in intensity minus the negative changes in intensity:

$$\lambda[t_j] = \lambda_0[t_j] - \sum_{s_l \in t_j^s} \Delta\lambda[(t_j, s_l)] + \sum_{s_l \in {}^s t_j} \Delta\lambda[(s_l, t_j)] \quad \forall\ t_j \in T \tag{4}$$

which can be expressed in matrix form as:

$$\lambda = \lambda_0 + Z_\lambda \Delta\lambda \tag{5}$$

where Z_λ is a matrix with rows indexed by T and columns indexed by E_S^T.

Example 4. *Equation* (5) *of the net in Fig. 1(a) is:*

$$\begin{pmatrix}\lambda[t_1]\\ \lambda[t_2]\\ \lambda[t_3]\\ \lambda[t_4]\end{pmatrix} = \begin{pmatrix}\lambda_0[t_1]\\ \lambda_0[t_2]\\ \lambda_0[t_3]\\ \lambda_0[t_4]\end{pmatrix} + \begin{pmatrix}0 & 0 & 0\\ 1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{pmatrix} \begin{pmatrix}\Delta\lambda[(s_1,t_2)]\\ \Delta\lambda[(s_2,t_3)]\\ \Delta\lambda[(s_3,t_4)]\end{pmatrix}$$

In order to account for partially known default intensities, λ_0 is assumed to be linearly constrained as:

$$J_\lambda \lambda_0 \leq K_\lambda \tag{6}$$

where J_λ and K_λ are real matrices of appropriate size. For instance, $\lambda_0[t_1]$ in Fig. 1(a) is constrained as $1 \leq \lambda_0[t_1] \leq 2$, and the default intensities of the other transitions is 0 (and hence are not specified in the Figure).

The combination of (2), (3), (5) and (6) leads to the state equations of intensity nets:

$$\begin{aligned} SE_{\mathcal{N}_S}(m, J_\lambda, K_\lambda) = \{(m, \mu_P, \mu_E, \Delta\lambda, \lambda) | & m = \mu_P + Y_m \mu_E \\ & C\Delta\lambda \leq D\mu_E \\ & \lambda = \lambda_0 + Z_\lambda \Delta\lambda \\ & J_\lambda \lambda_0 \leq K_\lambda\} \end{aligned} \tag{7}$$

The solutions of these equations contain all the potential states, $(m, \mu_P, \mu_E, \Delta\lambda, \lambda)$, of the intensity net for a given marking m and given matrices J_λ and K_λ. Different additional constraints can be added to (7) to model known system features. As an example, the constraint $\mu_P[p_i] = 0$ models the fact that all the tokens of place p_i must be active.

2.2. Event Nets

The intensity of a transition t_j is the speed at which the process modeled by t_j can perform its activities. The integral of the intensity $\lambda[t_j]$ over time is referred as the number of actions produced in t_j and is denoted as $\sigma[t_j]$. Let $\Delta\sigma[e](\tau)$ denote the number of actions produced in the intensity arc $e \in E_S^T$ until time τ:

$$\Delta\sigma(\tau) = \int_0^\tau \Delta\lambda(s)\, \mathrm{d}s \tag{8}$$

then, the number of actions produced in transitions is:

$$\sigma(\tau) = \lambda_0 \tau + Z_\lambda \Delta\sigma(\tau) \tag{9}$$

Event nets determine how the produced actions can be used to perform changes in the marking. An event net can be denoted as $T/V/P$ nets, which can be interpreted as: actions in transitions T produce and consume tokens in places P through event handlers V. More formally:

Definition 3 (Event net). *An event net is a tuple* $\mathcal{N}_V = (P, T, V, E_V, A, B)$ *where* (P, T, V, E_V) *is a tripartite graph determining the net structure and* (A, B) *are matrices determining the potential marking changes produced by the actions.*

Similarly to intensity nets, the set of vertices of the net is partitioned into three sets, P is the set of places, T is the set of transitions, and:

- $V = \{v_1, \ldots, v_k, \ldots\}$ is a set of $|V|$ event handlers.

The event handlers, which are depicted as dots, model the different ways in which the actions in the transitions can change the marking of places. The vertices of the net are connected by the edges in E_V. Each pair of vertices can be connected by at most one edge. The set E_V is partitioned into two sets E_V^P and E_V^T, where E_V^P is a set of directed edges, or simply *arcs*, connecting places to event handlers and vice versa, and E_V^T is a set of undirected edges, or simply *edges*, connecting transitions and event handlers. More formally:

- Every $e \in E_V^P$ is either an arc $e = (p_i, v_k)$ from a place p_i to a handler v_k, or an arc $e = (v_k, p_i)$ from a handler v_k to a place p_i.
- Every $e \in E_V^T$ is an edge $e = \{t_j, v_k\}$ connecting a transition t_j and a handler v_k.

As in the intensity net, connections among places and transitions are not allowed. The following notation will be used:

- ${}^{p}v_k$ denotes the input places of v_k, i.e. ${}^{p}v_k = \{p_i | (p_i, v_k) \in E_V^P\}$
- v_k^p denotes the output places of v_k, i.e. $v_k^p = \{p_i | (v_k, p_i) \in E_V^P\}$
- ${}^{v}p_i$ denotes the input handlers of p_i, i.e. ${}^{v}p_i = \{v_k | (v_k, p_i) \in E_V^P\}$
- p_i^v denotes the output handlers of p_i, i.e. $p_i^v = \{v_k | (p_i, v_k) \in E_V^P\}$
- ${}^{t}v_k$ denotes the transitions connected to v_k, i.e. ${}^{t}v_k = \{t_j | \{t_j, v_k\} \in E_V^T\}$
- t_j^v denotes the handlers connected to t_j, i.e. $t_j^v = \{v_k | \{t_j, v_k\} \in E_V^T\}$

The state of an event net accounts not only for markings and number of actions, but also for the marking changes and the execution of actions:

Definition 4 (State). *The state of an event net $\mathcal{N}_V$ is given by the tuple $(\sigma, a_T, a_E, \Delta m, m)$, where:*

- $\sigma \in \mathbb{R}_{\geq 0}^{|T|}$ *is a vector indexed by T where $\sigma[t_j]$ is the number of actions produced in t_j.*
- $a_T \in \mathbb{R}_{\geq 0}^{|T|}$ *is a vector indexed by T where $a_T[t_j]$ is the number of actions available in t_j.*
- $a_E \in \mathbb{R}_{\geq 0}^{|E_V^T|}$ *is a vector indexed by E_V^T where $a_E[\{t_j, v_k\}]$ is the number of actions of t_j executed by v_k.*
- $\Delta m \in \mathbb{R}_{\geq 0}^{|E_V^P|}$ *is a vector indexed by E_V^P where $\Delta m[(p_i, v_k)]$ is the number of tokens in p_i consumed by v_k, and $\Delta m[(v_k, p_i)]$ is the number of tokens in p_i produced by v_k.*
- $m \in \mathbb{R}_{\geq 0}^{|P|}$ *is the marking, i.e. a vector indexed by P where $m[p_i]$ is the number of tokens in p_i.*

Since actions need time to be produced, at the initial state it holds $\sigma = 0$, $a_T = 0$ and $a_E = 0$.

Notice that the number of actions produced in a transition is equal to the number of actions executed by the event handlers plus the number of available actions:

$$\sigma[t_j] = a_T[t_j] + \sum_{v_k \in t_j^v} a_E[\{t_j, v_k\}] \quad \forall\ t_j \in T \tag{10}$$

This can be expressed in matrix form as:

$$\sigma = a_T + Y_\sigma a_E \tag{11}$$

where Y_σ is a matrix with rows indexed by T and columns indexed by E_V^T.

Similarly to intensity handlers, each event handler $v_k \in V$ is associated with a set of linear inequalities that relate the number of actions executed in the transitions connected to it to the marking changes in the places connected to it. The coefficients of these inequalities can be captured by two matrices A and B of real numbers and same number of rows. The columns of A and B are indexed by E_V^P and E_V^T, respectively. The relation between executed actions, a_E, and marking changes, Δm, can be expressed by:

$$A\Delta m \leq B a_E \tag{12}$$

The number of tokens in a place p_i is equal to the initial number of tokens, which is denoted $m_0[p_i]$, minus the number of tokens consumed plus the number of tokens produced:

$$m[p_i] = m_0[p_i] - \sum_{v_k \in p_i^v} \Delta m[(p_i, v_k)] + \sum_{v_k \in {}^v p_i} \Delta m[(v_k, p_i)] \quad \forall\ p_i \in P \tag{13}$$

which can be expressed in matrix form as:

$$m = m_0 + Z_m \Delta m \tag{14}$$

where Z_m is a matrix with rows indexed by P and columns indexed by E_V^P.

Example 5. *Fig. 1(b) depicts an event net with* 4 *places,* 4 *transitions and* 5 *event handlers. As in the intensity net, labels are associated with edges and arcs to simplify the writing of inequalities. For instance, the equation a=x associated with* v_1 *means that each action of* t_1 *executed by* v_1 *produces one token in* p_1*. The inequalities associated with* v_2 *(written down on the left of the Figure for clarity) imply that each action of* t_2 *executed by* v_2 *consumes a token from* p_1 *and produces a number of tokens in the interval* $[2, 3]$ *in* p_2 *(the actual number of tokens produced in* p_2 *is chosen nondeterministically).*

In order to simplify the notation, no inequalities are written when the labels of all the edges and arcs of a handler are forced to be equal. For instance, the equation associated with v_3 *in Fig. 1(b) is a=x and hence it is omitted (the equations of* v_4 *and* v_5 *are also omitted). The same omission is done for event and intensity handlers in the following.*

Partially known initial marking can be accounted for by:

$$J_m m_0 \leq K_m \tag{15}$$

where J_m and K_m are real matrices of appropriate size.

Equations (11), (12), (14) and (15) compose the state equations of event nets:

$$\begin{aligned} SE_{\mathcal{N}_V}(\sigma, J_m, K_m) = \{(\sigma, a_T, a_E, \Delta m, m) | \sigma = a_T + Y_\sigma a_E \\ A\Delta m \leq B a_E \\ m = m_0 + Z_m \Delta m \\ J_m m_0 \leq K_m\} \end{aligned} \tag{16}$$

These equations represent necessary reachability conditions for the state of an event net, for a given number of produced actions, σ, and an initial marking constrained by the matrices J_m and K_m. The solution space can be further constrained by known system features. For instance, the constraint $a_T[t_j] = 0$ forces the execution of all the actions produced in t_j.

2.3. Flexible Nets

This section introduces Flexible Nets (FNs), which can be denoted as $P/H/T$ nets, i.e. places P and transitions T are connected by event and intensity handlers.

Definition 5 (FN). *A Flexible Net (FN) is a tuple* $\mathcal{N} = (P, T, V, E_V, A, B, S, E_S, C, D)$ *where* (P, T, V, E_V, A, B) *is an event net and* (P, T, S, E_S, C, D) *is an intensity net.*

Example 6. *Fig. 2 shows a FN composed by the intensity net in Fig. 1(a) and the event net in Fig. 1(b). In the resulting FN, the event net determines the way actions produce marking changes, and the intensity net determines the way tokens produce intensity changes.*

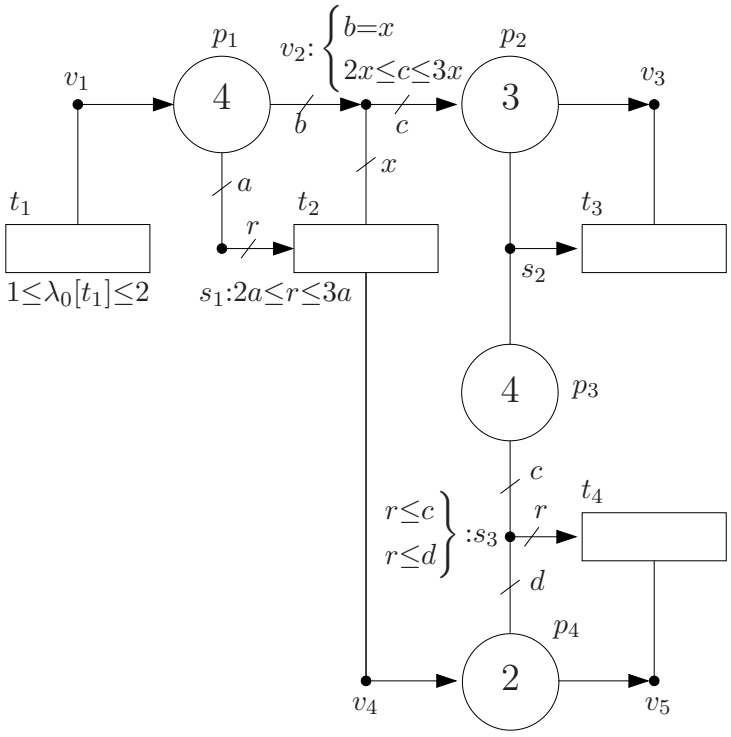

Figure 2: FN resulting of combining the intensity net and the event net in Fig. 1.

In addition to the state variables of the event and intensity net, $\Delta\sigma$ (see (8)) is included in the tuple of variables defining the state of the FN.

Definition 6 (State). *The state* $\mathbf{x}$ *of an FN is given by the vector that results from the concatenation of the state variables, i.e.* $\mathbf{x} = (m, \mu_P, \mu_E, \Delta\lambda, \lambda, \Delta\sigma, \sigma, a_T, a_E, \Delta m)$.

All the state variables are time dependent. For the sake of clarity, the time dependency will be omitted when it is clear from the context, e.g. $\sigma(\tau)$ is shortened to σ. In the initial state at time 0 it holds $\Delta\sigma = 0$, $\sigma = 0$, $a_T = 0$, $a_E = 0$, $\Delta m = 0$, i.e. the initial state can be written as: $(m, \mu_P, \mu_E, \Delta\lambda, \lambda, 0, 0, 0, 0, 0, 0)$.

By making use of $SE_{\mathcal{N}_S}(m, J_\lambda, K_\lambda)$ (see (7)), (8), (9), and $SE_{\mathcal{N}_V}(\sigma, J_m, K_m)$ (see (16)), it is possible to write a set of equations that any potential state at time τ must satisfy.

Proposition 1 (State equations (FN)). *Let $\mathcal{N}$ be an FN with initial marking m_0 satisfying $J_m m_0 \leq K_m$, and default intensities λ_0 satisfying $J_\lambda \lambda_0 \leq K_\lambda$. Every state $(m, \mu_P, \mu_E, \Delta\lambda, \lambda, \Delta\sigma, \sigma, a_T, a_E, \Delta m)$ reachable at time τ belongs to $SE_{\mathcal{N}}(\tau, J_m, K_m, J_\lambda, K_\lambda)$ where:*

$$
\begin{aligned}
SE_{\mathcal{N}}(\tau, J_m, K_m, J_\lambda, K_\lambda) = \{&(m, \mu_P, \mu_E, \Delta\lambda, \lambda, \Delta\sigma, \sigma, a_T, a_E, \Delta m) | \\
& m = \mu_P + Y_m \mu_E;\ C\Delta\lambda \leq D\mu_E;\ \lambda = \lambda_0 + Z_\lambda \Delta\lambda;\ J_\lambda \lambda_0 \leq K_\lambda \\
& \Delta\sigma = \int_0^\tau \Delta\lambda(s)\,\mathrm{d}s;\ \sigma = \lambda_0 \tau + Z_\lambda \Delta\sigma \\
& \sigma = a_T + Y_\sigma a_E;\ A\Delta m \leq B a_E;\ m = m_0 + Z_m \Delta m;\ J_m m_0 \leq K_m\}
\end{aligned}
\tag{17}
$$

where every variable is nonnegative.

Thus, an FN is a time continuous model, where time, denoted as τ, is the independent variable, and all the state variables are nonnegative reals.

Equations (17) can be interpreted as follows: At a given time τ, some of the produced actions (σ) are available (a_T), and the rest (a_E) were executed before τ. The executed actions produced marking changes (Δm) which resulted in the marking m in places at τ. Some of the tokens in m are active (μ_E) and the rest are idle (μ_P). Active tokens produce intensity changes ($\Delta\lambda$) which result in overall intensities (λ) in transitions at τ. The integral of the intensity changes and overall intensities over time after τ will produce more actions (σ), i.e. σ is produced as time elapses. This behavior repeats over time: when a new marking is reached, intensities are updated, which can lead to the production and execution of new actions, which consequently results in a new marking.

Equations (17) can be relaxed by dropping their time dependency. This leads to a set of constraints that represent a necessary condition for reachability at any time.

Proposition 2 (Untimed state equations (FN)). *Let $\mathcal{N}$ be an FN with initial marking m_0 satisfying $J_m m_0 \leq K_m$, and default intensities λ_0 satisfying $J_\lambda \lambda_0 \leq K_\lambda$. Every state $(m, \mu_P, \mu_E, \Delta\lambda, \lambda, \Delta\sigma, \sigma, a_T, a_E, \Delta m)$ reachable*

at any time $\tau \geq 0$ belongs to $USE_{\mathcal{N}}(J_m, K_m, J_\lambda, K_\lambda)$ where:

$$\begin{aligned} USE_{\mathcal{N}}(J_m, K_m, J_\lambda, K_\lambda) = \{(m, \mu_P, \mu_E, \Delta\lambda, \lambda, \Delta\sigma, \sigma, a_T, a_E, \Delta m)| \\ m = \mu_P + Y_m\mu_E;\ C\Delta\lambda \leq D\mu_E;\ \lambda = \lambda_0 + Z_\lambda\Delta\lambda;\ J_\lambda\lambda_0 \leq K_\lambda \\ \sigma = a_T + Y_\sigma a_E;\ A\Delta m \leq Ba_E;\ m = m_0 + Z_m\Delta m;\ J_m m_0 \leq K_m\} \end{aligned} \tag{18}$$

where every variable is nonnegative.

3. Guarded Flexible Nets

FNs can account for linear relationships between the marking of places and the intensities produced in transitions by means of the inequalities associated with intensity handlers. Although these inequalities allow for the modeling of a number of dynamical behaviors, they cannot accommodate the nonlinear dynamics of many systems of interest. This section introduces an extension of FNs, called guarded FNs (GFNs), that associates guards with the intensity arcs. In a GFN, intensity is produced at an intensity arc only if one of its guards is active. As it will be shown, GFNs are specially well suited to model hybrid systems.

3.1. Regions and partitions

The structure of a GFN, denoted $\mathcal{N}_G$, is defined in the same way as the structure of an unguarded one. The difference lies in the guards assigned to each intensity arc $e \in E_S^T$. Each guard is defined by a region of the state space, and a guard is said to be *active* if and only if the state of the net is in the region that defines that guard. A set of linear inequalities relating active tokens and intensities is associated with each guard of each arc. The intensity $\Delta\lambda[e]$ of $e \in E_S^T$ is determined by the set of linear inequalities associated with the guard of e that is active, if no guard is active then $\Delta\lambda[e] = 0$.

More formally, each region $\mathcal{R}_r$ is a convex polytope of the form $\mathcal{R}_r = \{\mathbf{y} \mid S_r\mathbf{y} \leq Q_r\}$ where $S_r(Q_r)$ is a real valued matrix(vector) and the components of $\mathbf{y}$ are associated with the state variables as $\mathbf{y} = (m, \mu_P, \mu_E, \Delta\sigma, \sigma, a_T, a_E, \Delta m)$. Hence, only the state variables in $\mathbf{y}$ can be used to define guards (notice that such guards will in turn determine the value of the intensities, and thus, neither $\Delta\lambda$ nor λ are used to define guards). The set of all the regions is denoted $\mathcal{R}$.

For modeling and analysis purposes, it is useful to consider sets of regions that partition the state space. A partition $\mathcal{P}_n$ is a set of regions $\mathcal{P}_n = \{\mathcal{R}_1, \ldots, \mathcal{R}_r, \ldots\}$ such that $\mathcal{R}_r \cap \mathcal{R}_s = \emptyset$ for every pair of regions $\mathcal{R}_r, \mathcal{R}_s \in \mathcal{P}_n$, and all the values of the state variables in $\mathbf{y}$ that satisfy (18) are contained in $\bigcup_{\mathcal{R}_r \in \mathcal{P}_n} \mathcal{R}_r$. In order to allow the modeler to partition the state space in

different ways, a set of partitions can be considered. The set of partitions is denoted $\mathcal{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_n, \ldots\}$.

The guards and the intensities determined by the guards can be included in the graphical representation of the net by associating linear inequalities and boolean conditions (that represent the guards) with intensity handlers.

Example 7. *Let us associate the following inequalities with the intensity handlers s_1 and s_3 of the FN in Fig. 2:*

$$s_1 : \begin{cases} 2a \leq r \leq 3a \text{ if } a \leq 3 \\ r = 4a \text{ otherwise} \end{cases} \qquad s_3 : \begin{cases} r = c \text{ if } c \leq d \\ r = d \text{ otherwise} \end{cases} \tag{19}$$

Such association turns the net in Fig. 2 into a GFN. The inequalities of s_1 mean that if the number of active tokens in $\{p_1, s_1\}$, $\mu_E[\{p_1, s_1\}]$, which is shortened as a, is less than or equal to 3 then the intensity in (s_1, t_1) is in the interval $[2a, 3a]$, otherwise the intensity is equal to $4a$. The inequalities of s_3 imply that the intensity in (s_3, t_4) is equal to the minimum number of active tokens in $\{p_3, s_3\}$ and $\{p_4, s_3\}$.

The above inequalities represent two partitions, $\mathcal{P}_1$ and $\mathcal{P}_2$, of the state space. Partition $\mathcal{P}_1$ contains 2 regions, $\mathcal{P}_1 = \{\mathcal{R}_1, \mathcal{R}_2\}$, where $\mathcal{R}_1$ is defined by $\mu_E[\{p_1, s_1\}] \leq 3$, and $\mathcal{R}_2$ by $\mu_E[\{p_1, s_1\}] \geq 3$. Partition $\mathcal{P}_2$ also contains 2 regions, $\mathcal{P}_2 = \{\mathcal{R}_3, \mathcal{R}_4\}$, where $\mathcal{R}_3$ is defined by $\mu_E[\{p_3, s_3\}] \leq \mu_E[\{p_4, s_3\}]$, and $\mathcal{R}_4$ is defined by $\mu_E[\{p_3, s_3\}] \geq \mu_E[\{p_4, s_3\}]$.

Notice that the non-strict inequalities used to define regions entails a non-null intersection of the polytopes at the borders. When the state of the net is at a shared border, it will be assumed to be in only one of the regions (any of them) sharing that border.

3.2. Definition and state equations

In a guarded net, each intensity arc $e \in E_S^T$ is assigned a set of regions through the function $\varphi : E_S^T \to 2^{\mathcal{R}}$. Each region $\mathcal{R}_r \in \varphi(e)$ is a guard of e that is denoted as g_r. It is assumed that the regions in $\varphi(e)$ are disjoint, i.e. $\mathcal{R}_r \cap \mathcal{R}_s = \emptyset$ for every pair of regions $\mathcal{R}_r, \mathcal{R}_s \in \varphi(e)$. Thus, at most, one guard of a given intensity arc is active at any given time.

In order to account for intensities that can be determined by different sets of linear inequalities, two vectors are considered for the intensity of arcs: $\Delta\lambda_U(\tau)$ and $\Delta\lambda(\tau)$.

The vector $\Delta\lambda_U(\tau)$ contains all the potential intensities that the arcs can have, hence, a component of $\Delta\lambda_U$ corresponds to an arc $e \in E_S^T$ and a guard of e, thus, $\Delta\lambda_U$ is indexed by the pairs (e, g_r) where $e \in E_S^T$ and $\mathcal{R}_r \in \varphi(e)$.

Notice that the value of $\Delta\lambda_U[(e, g_r)]$ can be negative if the state is not in $\mathcal{R}_r$. In a guarded net, $\Delta\lambda_U$ is considered a state variable, therefore, the state of a guarded net is given by $\mathbf{x} = (m, \mu_P, \mu_E, \Delta\lambda_U, \Delta\lambda, \lambda, \Delta\sigma, \sigma, a_T, a_E, \Delta m)$.

The vector $\Delta\lambda(\tau)$ represents the actual intensity in arcs. Hence, as in unguarded nets, $\Delta\lambda$ is indexed by $e \in E_S^T$ and is nonnegative.

In a guarded net, matrices A and B are defined as in unguarded nets, and matrices C and D are used to determine $\Delta\lambda_U$. More precisely, the value of $\Delta\lambda_U$ can be determined by:

$$C\Delta\lambda_U \leq D\mu_E \tag{20}$$

where C and D are defined in such a way that every pair (e, g_r) is taken into account.

Example 8. *Let us consider the partitions, regions and inequalities of the GFN defined in Example 7. The guards associated with the intensity arc (s_1, t_2) are g_1 and g_2; and the guards associated with (s_3, t_4) are g_3 and g_4. Notice that the intensity arc (s_2, t_3) is not guarded and, hence, its intensity is produced as in an unguarded FN. Equation* (20) *of the GFN is:*

$$\begin{pmatrix} -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \Delta\lambda_U[((s_1,t_2),g_1)] \\ \Delta\lambda_U[((s_1,t_2),g_2)] \\ \Delta\lambda_U[(s_2,t_3)] \\ \Delta\lambda_U[((s_3,t_4),g_3)] \\ \Delta\lambda_U[((s_3,t_4),g_4)] \end{pmatrix} \leq \begin{pmatrix} -2 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 \\ -4 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \mu_E[\{p_1,s_1\}] \\ \mu_E[\{p_2,s_2\}] \\ \mu_E[\{p_3,s_2\}] \\ \mu_E[\{p_3,s_3\}] \\ \mu_E[\{p_4,s_3\}] \end{pmatrix}$$

A GFN is defined as follows:

Definition 7 (GFN). *A Guarded Flexible Net (GFN) is a tuple $\mathcal{N}_G = (P, T, V, E_V, A, B, S, E_S, C, D, \mathcal{P}, \varphi)$ where $(P, T, V, E_V, A, B, S, E_S)$ denote the same elements as in an FN, C and D account for the all the potential intensities in arcs, $\mathcal{P}$ is a set of partitions, and φ is a function that associates regions with intensity arcs.*

In order to establish state equations, let us define a binary variable $\delta_r(\tau)$ per region $\mathcal{R}_r$ that indicates whether the state is in $\mathcal{R}_r$:

$$\delta_r(\tau) = \begin{cases} 1 & \text{if } \mathbf{x}(\tau) \in \mathcal{R}_r \\ 0 & \text{otherwise} \end{cases} \tag{21}$$

Let $e \in E_S^T$ and $\mathcal{R}_r \in \varphi(e)$, if the state is in $\mathcal{R}_r$ then the intensity $\Delta\lambda[e](\tau)$ is $\Delta\lambda_U[(e, g_r)](\tau)$. Thus, $\Delta\lambda$ can be expressed in matrix form as:

$$\Delta\lambda(\tau) = \delta(\tau)\Delta\lambda_U(\tau) \tag{22}$$

where $\delta[e,(e,g_r)](\tau) = \delta_r(\tau)$ (the pair (e, g_r) is the index of the column associated with the guard g_r of e) for every $e \in E_S^T$ and every $\mathcal{R}_r \in \varphi(e)$, and the rest of the elements of δ are 0.

Example 9. *Equation* (22) *for the guarded net described in Example 7 is:*

$$\begin{pmatrix} \Delta\lambda[(s_1,t_2)] \\ \Delta\lambda[(s_2,t_3)] \\ \Delta\lambda[(s_3,t_4)] \end{pmatrix} = \begin{pmatrix} \delta_1 & \delta_2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \delta_3 & \delta_4 \end{pmatrix} \begin{pmatrix} \Delta\lambda_U[((s_1,t_2),g_1)] \\ \Delta\lambda_U[((s_1,t_2),g_2)] \\ \Delta\lambda_U[(s_2,t_3)] \\ \Delta\lambda_U[((s_3,t_4),g_3)] \\ \Delta\lambda_U[((s_3,t_4),g_4)] \end{pmatrix}$$

where, for instance, $\delta_1 = 1$ *iff* $\mu_E[\{p_1, s_1\}] \leq 3$.

Equation (22) can be used to write the state equations of a GFN.

Proposition 3 (State equations (GFN)). *Let* $\mathcal{N}_G$ *be a GFN with initial marking* m_0 *satisfying* $J_m m_0 \leq K_m$*, and default intensities* λ_0 *satisfying* $J_\lambda \lambda_0 \leq K_\lambda$*. Every state* $(m, \mu_P, \mu_E, \Delta\lambda_U, \Delta\lambda, \lambda, \Delta\sigma, \sigma, a_T, a_E, \Delta m)$ *reachable at time* τ *belongs to* $GSE_{\mathcal{N}_G}(\tau, J_m, K_m, J_\lambda, K_\lambda)$ *where:*

$$\begin{aligned} GSE_{\mathcal{N}_G}(\tau, J_m, K_m, J_\lambda, K_\lambda) = \{&(m, \mu_P, \mu_E, \Delta\lambda_U, \Delta\lambda, \lambda, \Delta\sigma, \sigma, a_T, a_E, \Delta m) | \\ & m = \mu_P + Y_m\mu_E;\ C\Delta\lambda_U \leq D\mu_E;\ \Delta\lambda = \delta\Delta\lambda_U; \lambda = \lambda_0 + Z_\lambda\Delta\lambda;\ J_\lambda\lambda_0 \leq K_\lambda \\ & \Delta\sigma = \int_0^\tau \Delta\lambda(s)\,\mathrm{d}s;\ \sigma = \lambda_0\tau + Z_\lambda\Delta\sigma \\ & \sigma = a_T + Y_\sigma a_E;\ A\Delta m \leq Ba_E;\ m = m_0 + Z_m\Delta m;\ J_m m_0 \leq K_m\} \end{aligned} \tag{23}$$

where every variable, except $\Delta\lambda_U$*, is nonnegative.*

In [18], a set of constraints that represent necessary reachability conditions, i.e. that contains all the solutions of (23), was developed. Such constraints consist of linear and quadratic inequalities that combine real and binary variables. The combination of such constraints with an objective function results in a programming problem that can be used to compute bounds on the state variables at a given time τ. A series of intermediate states at different time instants can be considered to obtain time trajectories.

4. Modeling and analysis of hybrid systems

Some of the modeling capabilities of GFNs and different types of analyses that can be conducted by employing these nets are presented in this section. The plots in the rest of the paper have been carried out by the tool `fnyzer` (available at `https://bitbucket.org/Julvez/fnyzer.git`) which builds a programming problem from a GFN and an objective function [18]. The tool makes use of Pyomo [22] to build the programming problems and Gurobi [23] and CPLEX [24] to solve them. The `fnyzer` tool was executed on a desktop computer (Intel i7, 2.00 GHz, 8 GiB, Ubuntu 14.04 LTS).

The net in Fig. 3(a) represents a simple cycle of two places p_1 and p_2 whose tokens are consumed and produced by two event handlers v_1 and v_2. These event handlers make use of the actions in t_1 and t_2. In this net, all the tokens are forced to be active, and all the actions are forced to be executed. Hence, given that the intensity arc (s_2, t_2) is unguarded, the rate at which actions are produced in t_2, i.e. $\lambda[t_2]$, is equal to $m[p_2]$, and $\lambda[t_2]$ is also the rate at which tokens are consumed from p_2 and produced in p_1. The arc (s_1, t_1) has two guards, one is defined by the region $m[p_1]{\leq}5$ and the other by the region $m[p_1]{\geq}5$. The inequalities associated with the guards state that $\lambda[t_1] = 2m[p_1]$ if $m[p_1]{\leq}5$, and $1.4{\leq}\lambda[t_1]{\leq}1.6$ otherwise. The initial marking is $m_0[p_1]{=}6$ and $m_0[p_2]{=}0$.

Given that the intensity of t_1 is uncertain (but constrained to $[1.4, 1.6]$) if $m[p_1]{\geq}5$, different time trajectories of the state are possible. Fig. 3(b) shows the time trajectory of $m[p_1]$ when $m[p_1]$ is minimized(red trajectory) and maximized(blue trajectory). Notice that, given the structure of the event net, $m[p_2]$ can be computed by $m[p_2]{=}6{-}m[p_1]$ at any time instant. The time trajectories have been obtained by a model predictive control (MPC) approach [15], according to which a number of time intervals (or sample times) are considered, and the programming problem mentioned above yields the state of the net for each sample time. At the end of the first interval the state of the system is updated and the procedure is repeated. In Fig. 3, the sample time was set to 0.1 time units and the prediction horizon to one step. The dotted line is the border between the two regions. The green trajectory is obtained by setting $\lambda[t_1]{=}1.5$, i.e. there is no uncertainty, and the state evolution is given by a system of ordinary differential equations (ODE).

Let the regions $m[p_1]{\leq}5$ and $m[p_1]{\geq}5$ be denoted as L and U respectively. Initially the system is in region U. In order to minimize(maximize) $m[p_1]$, the solver sets $\lambda[t_1]$ to 1.6(1.4). Thus, the trajectory switches from region U to region L at different time instants depending on whether $m[p_1]$ is minimized or maximized. Notice that the obtained upper and lower bounds envelop the ODE trajectory when $\lambda[t_1]$ is set to 1.5. Hence, GFNs can be used to com-

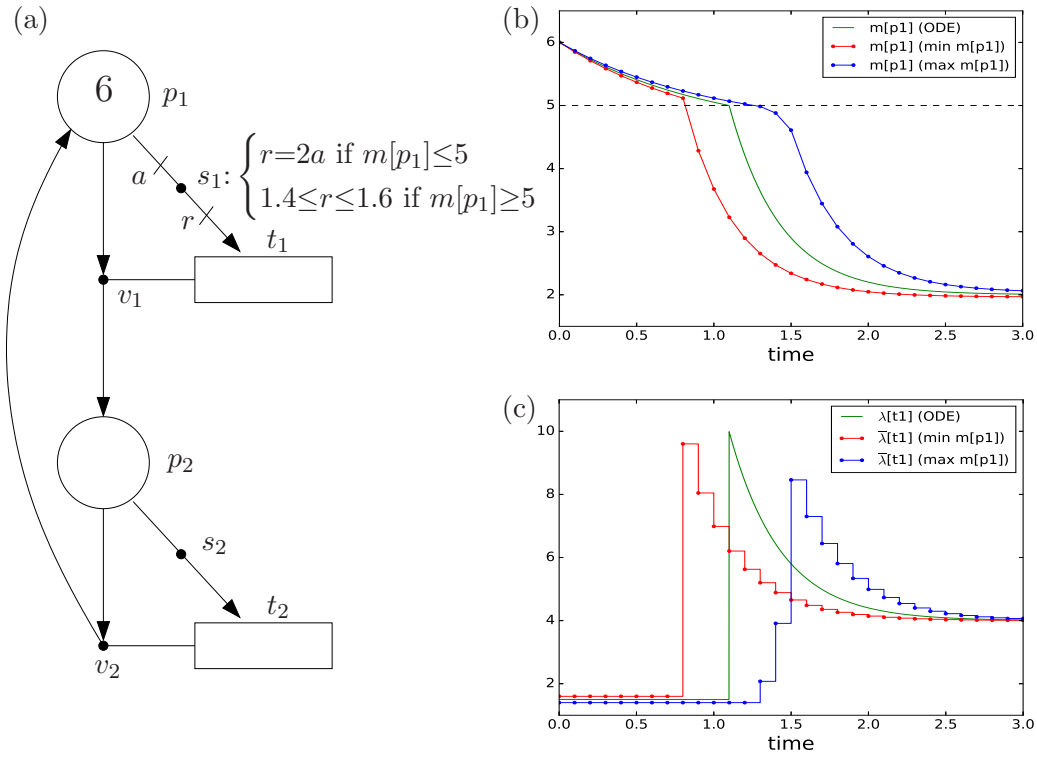

Figure 3: (a) GFN with two regions. (b) Time trajectory of $m[p_1]$. (c) Time trajectory of $\bar{\lambda}[t_1]$.

pute trajectories that bound the potential evolutions of dynamical systems with uncertainties, and thus, that are difficult to integrate. The tightness of the bounds can be improved by decreasing the sample time. The average intensity of t_1, $\bar{\lambda}[t_1]$, at each sample time when $m[p_1]$ is minimized(maximized) is the red(blue) trajectory in Fig. 3(c). The time trajectory of the intensity of t_1 obtained by the ODE with $\lambda[t_1]$=1.5 for the region U is shown in green. At the steady state, the equality $\bar{\lambda}[t_1]$=$\bar{\lambda}[t_2]$=4 necessarily holds true. The CPU time to compute each interval of the MPC was 5.34s.

In guarded nets, it is useful to consider a binary variable α_r that indicates if the region $\mathcal{R}_r$ has been visited and has contributed to the system dynamics. For instance, for the red trajectory in Fig. 3(b) (when $m[p_1]$ is minimized), $\alpha_L = 0$ and $\alpha_U = 1$ until time interval $[0.8, 0.9]$, i.e. the system dynamics of all the time intervals before time 0.8 were determined exclusively by the region U. During the time interval $[0.8, 0.9]$, $\alpha_L = \alpha_U = 1$ because the overall system dynamics was a combination of the dynamics of both regions during this time interval (the system is crossing the border between regions). From time instant 0.9 onwards, $\alpha_L = 1$ and $\alpha_U = 0$.

More formally, the binary variable $\alpha_r \in \{0, 1\}$ is defined as:

$$\alpha_r = 0 \;\leftrightarrow\; \bar{\delta}_r = 0 \tag{24}$$

where $\bar{\delta}_r$ is the time ratio spent by the net in $\mathcal{R}_r$. More precisely:

$$\bar{\delta}_r = \frac{1}{\theta} \int_0^\theta \delta_r(\tau)\, \mathrm{d}\tau \tag{25}$$

where θ is the length of the considered time interval.

The following example shows that α_r can be used to force the dynamics of the system to be determined just by one region during each time interval.

The intensity arc (s_1, t_1) in the net in Fig. 4 has three guards. Let us denote the regions associated with the guards as: L ($m[p_2]{\leq}8$), M ($8{\leq}m[p_2]{\leq}15$), and U ($15{\leq}m[p_2]$). Thus, the intensity produced in (s_1, t_1), and hence in t_1, depends on the guard that is active, and is determined by the equations written next to s_1. Note that this relatively complex dynamical behaviour, which would require a more involved graphical representation with self-loops if Petri nets were used, can be expressed rather cleanly by the intensity net of the GFN. The intensities of t_2 and t_3 are constant and equal to 12 and 10 respectively. In this GFN, all the tokens are forced to be active, and all the actions are forced to be executed. Hence, the intensity in t_4 is equal to $m[p_1]$. The initial marking is $m_0[p_1] = 4$ and $m_0[p_2] = 12$.

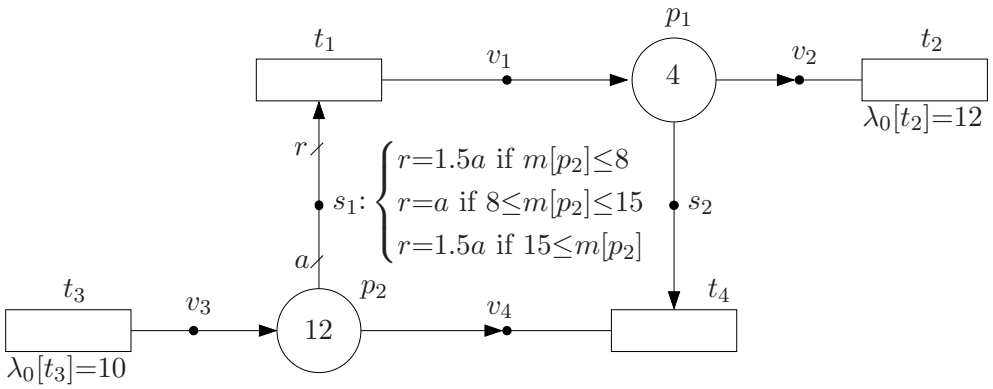

Figure 4: GFN with three regions.

Let us make use of the introduced binary variable α_r to force the system dynamics to be determined just by one region during each interval. This can be done by including the equation $\alpha_L + \alpha_M + \alpha_U = 1$ in the set of constraints of each time interval. In this way, the evolution of the state can be seen as that of a discrete time system in which the same linear dynamics are applied during each time interval.

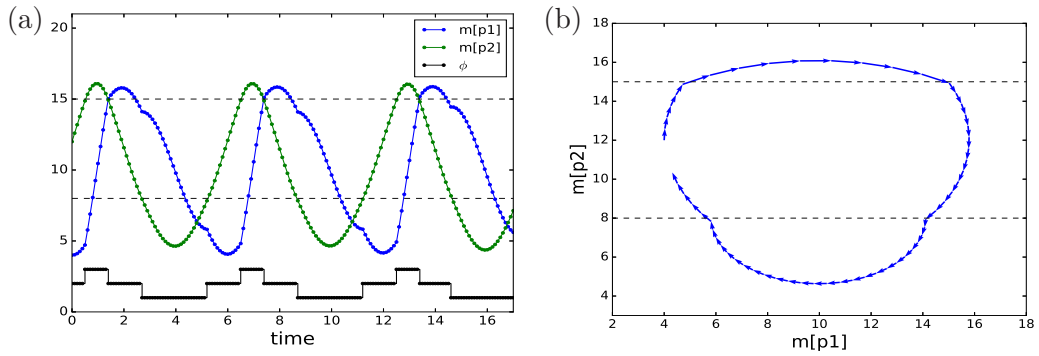

Figure 5: Time evolution (a) and evolution in the phase space (b) of the first cycle of the net in Fig. 4.

The time evolution of the net in Fig. 4 and its evolution in the phase space are shown in Fig. 5. As in the previous example, the trajectories are obtained through an MPC approach with sample time of 0.1 time units and prediction horizon of one step. The objective function is to minimize $m[p_1]$. Let us define a binary variable, ϕ, as $\phi = \alpha_L + 2\alpha_M + 3\alpha_U$, i.e. $\phi = 1(\phi = 2)(\phi = 3)$ if the net is in region $L(M)(U)$. The value of ϕ, which is depicted in the black trajectory in Fig. 5(a), can be interpreted as the discrete state of the system that determines the continuous dynamics. The dotted lines represent the borders of the regions. The CPU time to compute each interval of the MPC was $4.83s$.

Notice that GFNs can mimic the behavior of continuous Petri nets under different server semantics [25], e.g. t_2 in Fig. 3(a) is under infinite server semantics, and t_2 in Fig. 4 is under finite server semantics. With respect to hybrid Petri nets [11] (which include discrete and continuous Petri nets), it should be noted that the range of potential marking evolutions arising from the inequalities associated with handlers in GFNs, see Fig. 3(b), cannot be accommodated in a single hybrid Petri net. On the other hand, given that the marking evolution of a GFN is continuous, the marking discontinuities produced by discrete firings of hybrid Petris nets cannot be captured by the marking of GFNs.

5. Control of hybrid systems

As shown in the previous sections, GFNs can accommodate different types of uncertain parameters. If such uncertain parameters are thought of as control actions that can be applied to the system, the same approach to compute bounds of the system trajectory can be used to compute control actions that optimize a given control goal. This section demonstrates the

ability of GFNs to model and solve different control problems.

Let us focus on a GFN that models a hybrid system in which the dynamics of the continuous variables are determined by the discrete state, which in turn can be controlled by a delayed input action.

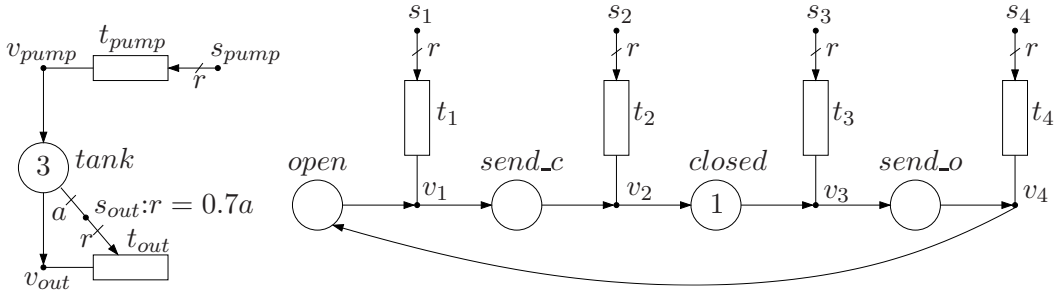

Figure 6: FN modeling a tank with a delayed control signal.

The net in Fig. 6 models a tank (place $tank$) from which liquid is removed by t_{out} at a rate $0.7m[tank]$, where $m[tank]$ is the level of liquid in the tank. The tank can be filled by means of a pump (transition t_{pump}) which can be either open or closed. If the pump is open, the tank is fed at a constant flow rate of 4.0, and if the pump is closed there is no flow into the tank. It is assumed that a delay of 0.5 time units exists from the moment the controller sends a signal to open (or close) the pump until the signal is received by the pump, which is then opened (or closed) instantaneously. The pump is initially closed and the initial level in the tank is $m_0[tank] = 3.0$.

The elements on the left of Fig. 6, i.e. $tank$, t_{pump}, t_{out} and the handlers connected to them, model the tank together with its input and output flows. The elements on the right are used to model the following discrete states: 1) pump open and no signal sent; 2) pump open and closing signal sent; 3) pump closed and no signal sent; 4) pump closed and opening signal sent. These states will be denoted as *open*, *send_close*, *closed*, and *send_open*, respectively. A region is associated with each of these possible states, the constraints that define the regions are: r_open: $m[open] \geq 1$; r_send_close: $\epsilon \leq m[send\_c], m[send\_o] = 0$; r_closed: $m[closed] \geq 1$ and r_send_open: $\epsilon \leq m[send\_o], m[send\_c] = 0$, where ϵ, set as $\epsilon = 10^{-3}$, is used to check if a marking is strictly positive. The initial marking of the places on the right is $m_0[closed] = 1$, $m_0[open] = m_0[send\_c] = m_0[send\_o] = 0$, i.e. the pump is initially closed.

In this net the sum of α_r (see (24)) is forced to be 1, i.e. $\alpha_{open} + \alpha_{send\_close} + \alpha_{closed} + \alpha_{send\_open} = 1$. This constraint has the following implications: a) during a given time interval, the system is driven by the dynamics of just one region; and b) the system state is forced to remain within the constraints

that define the regions, i.e. it cannot move away from those constraints (this implies that $m[send\_o]$ and $m[send\_c]$ cannot be strictly positive at the same time as such a state would not belong to any of the four defined regions).

The intensity of each transition, except t_{out} that satisfies $\lambda[t_{out}] = 0.7m[tank]$, is determined by the region at which the state lies as follows:

$$s_{pump}: \begin{cases} r{=}4.0 \ if \ r\_open \ or \ r\_send\_close \\ r{=}0 \ otherwise \end{cases}$$

$$s_1: \begin{cases} 0{\leq}r{\leq}M \ if \ r\_open \ or \ r\_send\_close \\ r{=}0 \ otherwise \end{cases}$$

$$s_2: \begin{cases} r{=}ss \ if \ r\_send\_close \\ r{=}0 \ otherwise \end{cases}$$

$$s_3: \begin{cases} 0{\leq}r{\leq}M \ if \ r\_closed \ or \ r\_send\_open \\ r{=}0 \ otherwise \end{cases}$$

$$s_4: \begin{cases} r{=}ss \ if \ r\_send\_open \\ r{=}0 \ otherwise \end{cases}$$

where M and ss are constants used to model the time it takes for a signal to reach the pump. Namely, M is an arbitrarily high value set to $M = 10.0$ to upper bound the speeds of t_1 and t_3, and ss represents the speed of the signal, which is set to $ss = 2.0$ to model the 0.5 time units delay of the signal. In words, assume that $m[open] = 1$, i.e. the pump is open and no signal has been sent, and the controller decides to send a closing signal at time τ, which implies moving to region r_send_close. The token in *open* can then move at maximum rate $M = 10.0$ to $send\_c$ and exactly at rate $ss = 2.0$ to *closed* so that at time τ+0.5 the equality $m[closed] = 1$ will hold true. The same mechanism is used for the signal to open the pump. In this net, only the tokens in *tank* are forced to be active, and the actions of all the transitions are forced to be executed.

Let us assume that the control goal is to minimize the actuation of the pump, i.e. to minimize the number of times the pump is opened and closed, while the level of the tank is maintained within the interval $[1, 4]$. Thus, the controller must send the open and close signals at appropriate time instants so that the constraints on the level of the tank are not violated. This control goal is captured by the objective function

$$min \ \bar{\delta}_{send\_close} + \bar{\delta}_{send\_open}$$

where $\bar{\delta}_r$ denotes the time ratio spent in region $\mathcal{R}_r$ (see (25)) during the prediction horizon.

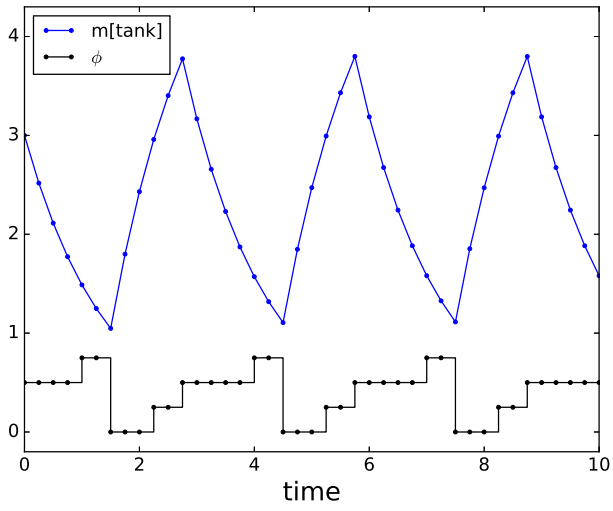

Figure 7: Control and marking trajectory of the net in Fig. 6.

Let the sample time be 0.25 time units; thus, the signal needs 2 sample times to reach the pump. Hence, in order to allow the controller to send the control signal sufficiently in advance, a prediction horizon of 3 sample times is used. Fig. 7 shows the trajectories of the level of the tank and the control required to minimize the objective function obtained by MPC. The control is represented by function ϕ, which is defined as $\phi = 0\alpha_{open} + 0.25\alpha_{send\_close} + 0.5\alpha_{closed} + 0.75\alpha_{send\_open}$. The pump is initially closed and therefore the level of liquid in the tank decreases. After four time intervals, i.e. at time 1.0, the signal to open the pump is sent by the controller. The pump is effectively opened at time 1.5. Then, it remains opened for 3 time periods, after which the close signal is sent. Since the goal was to minimize the number of times the pump was actuated, the level of liquid in the tank gets close to the set limits 1 and 4. The CPU time to compute each interval of the MPC was $7.94s$.

Let us now explore how GFNs can model a discrete control action and how sample intervals of different lengths can be used to improve the performance of the controller.

The net in Fig. 8(a) models a system in which the intensity of t_1, $\lambda[t_1]$, can be either 0.5 or 3.0. The actual value of $\lambda[t_1]$ depends on the number of active tokens in $\{p_u, s_1\}$, $\mu_E[(p_u, s_1)]$; if $\mu_E[(p_u, s_1)] \leq 0.5$ then $\lambda[t_1] = 0.5$, else $\lambda[t_1] = 3.0$. Thus, the number of active tokens in $\{p_u, s_1\}$ acts a as switch that determines $\lambda[t_1]$. The token in p_u is not forced to be active, and hence, $\mu_E[(p_u, s_1)]$ becomes a control variable that can be used to optimize a given control objective. The region defined by $\mu_E[(p_u, s_1)] \leq 0.5$ will be called

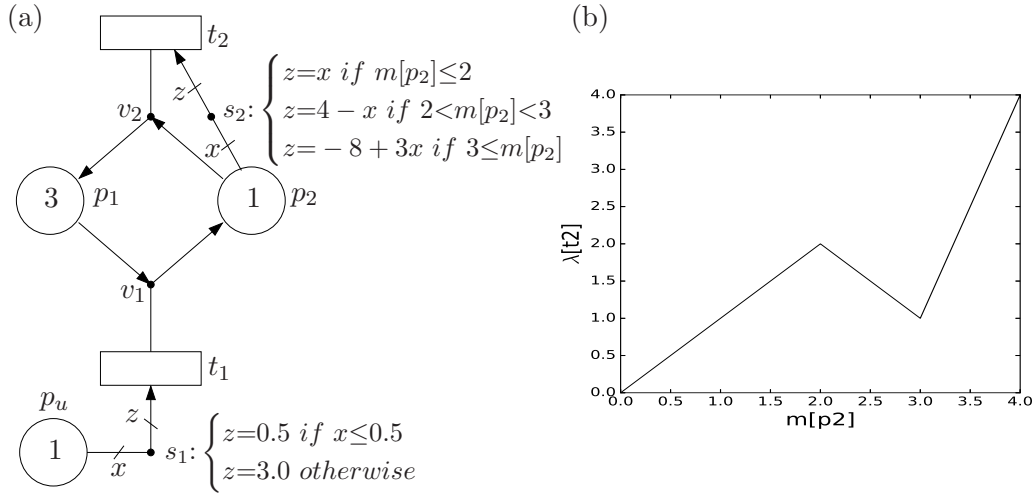

Figure 8: (a) GFN with a discrete control action modeled by the active tokens in $\{p_u, s_1\}$. (b) Piecewise linear function associated with $\lambda[t_2]$.

OFF, and the region $\mu_E[(p_u, s_1)] \geq 0.5$ will be called ON. It is assumed that the system can commute between the ON and OFF regions at most every 0.1 time units, i.e. once the system is in one region, it cannot move to the other until at least 0.1 time units have elapsed. The intensity of t_2 is a piecewise linear function of $m[p_2]$, see Fig. 8(b). All the tokens in p_2 are forced to be active and all the actions in the transitions are forced to be executed.

An MPC approach is considered where the control goal is to maximize the average intensity of t_2, $\bar{\lambda}[t_2]$, during the prediction horizon. The final time is set to 3.5. Two approaches to define prediction horizons are assessed:

a) *Even time intervals*: The prediction horizon consists of two time intervals, each interval of 0.1 time units. In order to model the requirement that the gap between switches must be at least 0.1, the system is forced to be at only one region of each partition during each interval.

b) *Uneven time intervals*: The prediction horizon consists of two intervals, the length of the first interval is 0.1 time units, and the second interval spans until the final time 3.5. For instance, at time 0, the first interval is $[0.0, 0.1]$ and the second interval is $[0.1, 3.5]$; at time 0.1, the first interval is $[0.1, 0.2]$ and the second interval is $[0.2, 3.5]$. The system is forced to be at only one region of each partition only during the first interval. Thus, the system evolution during the second interval can be the result of combining the dynamics of more than one region. In particular, this allows the control action to be ON only for some time in the potentially long second interval, and the average intensity of t_2 can be the result of combining several segments of the piecewise linear function associated with s_2.

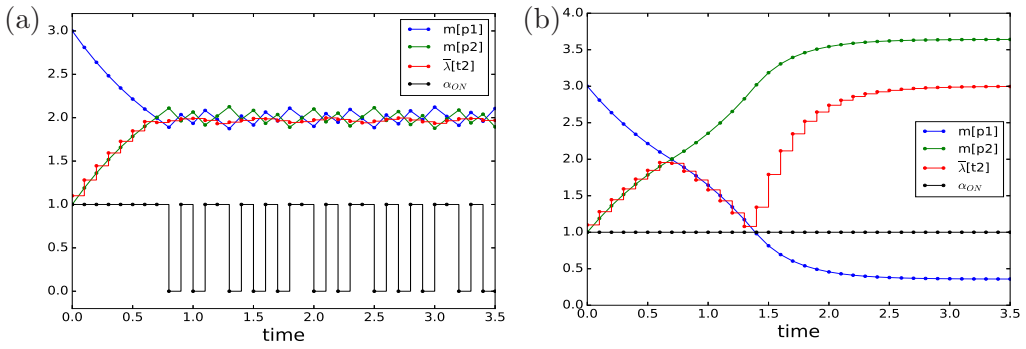

Figure 9: Time evolution of markings, $\bar{\lambda}[t_2]$ and control action of Fig. 8 under constant (a), and variable length (b) of the second interval of the prediction horizon.

Figs. 9(a) and Figs. 9(b) show the evolution of the system with the prediction horizons described in a) and b) respectively, where α_{ON} is equal to 1(0) if the system is in region ON(OFF), and hence $\lambda[t_1] = 3.0(\lambda[t_1] = 0.5)$.

In approach a), the action is ON until time instant 0.8. At this instant $m[p_2]$ is close to 2.0 and a local maximum for $\lambda[t_2]$ is reached, see Fig. 8(b). From time 0.8 the control action switches from ON to OFF in order to keep $m[p_2]$ as close to 2.0 as possible. The average time spent on the ON action is 0.6, which yields $\bar{\lambda}[t_1] = 2.0$.

In approach b), the control action is always ON. As in a), the local maximum is reached when $m[p_2]$ is close to 2.0, but given that the second interval extends up to time 3.5, higher values of the objective function can be expected if $m[p_2]$ is increased further. By keeping the action always ON, $m[p_2]$ and $\bar{\lambda}[t_2]$ tend asymptotically to 11/3 and 3.0, respectively. Thus, a higher value could be attained for the control objective with respect to approach a). A long second interval, which can be handled straightforwardly by GFNs, has the potential to stretch the prediction horizon without adding more intervals; hence, it can avoid "falling" to a local maximum without increasing excessively the computational burden. The CPU time to compute each interval of the MPC was $7.81s$ for approach a), and $10.35s$ for approach b). In addition to considering long intervals to tame the computational burden required to solve the associated programming problem, neighbor regions of the net could be merged into a single region that abstracts their dynamic behavior by bounding the intensities that can be produced. This will result in a programming problem with fewer binary variables, and hence, a lower computational burden at the cost of a lower accuracy, i.e. a less tight bound yielded by the solver. In this way, the trade-off "computational burden vs. accuracy" can be easily tuned in GFNs.

6. Conclusions

This paper describes the use of GFNs for the modeling, analysis and control of hybrid systems. In a GFN, the set of inequalities that determines the intensity in an arc depends on the current state of the net. More precisely, the state space is partitioned into regions, that are taken as guards of the intensity arcs, and each region is associated with a set of inequalities. Thus, on the one hand, the continuous state variables determine the region in which the state is and, on the other hand, such a region determines the dynamics of the continuous variables. This interplay between continuous and discrete states can be used to model a number of features of hybrid systems. All the potential trajectories that are consistent with the inequalities are accounted for by a set of constraints that represent necessary reachability conditions of the system. These constraints can be used to compute bounds for a given function of interest by a programming problem that includes such a function as its objective.

The same approach to compute bounds can be used to compute control actions that optimize a given function. In a GFN, different elements of the net, such as the default intensities or the number of active tokens, can be considered as control actions. The solution of the programming problem associated with a given objective function contains the values of the control actions that must be implemented on the system to control it in the desired way. GFNs have been demonstrated to be especially well suited for MPC approaches. In particular, they have been demonstrated to easily cope with delayed control actions, discrete control actions and uneven sample times.

Acknowledgements

This work was supported by the European Commission through a 7^{th} Framework Program BIOLEDGE Contract No: 289126 to SGO, and a Marie Curie Intra European Fellowship to JJ (FormalBio Contract No: 623995, Call reference: FP7-PEOPLE-2013-IEF). Further support came from the Biotechnology & Biological Sciences Research Council (UK) grant no. BB/N02348X/1 as part of the IBiotech Program, and by the Industrial Biotechnology Catalyst (Innovate UK, BBSRC, EPSRC) to support the translation, development and commercialisation of innovative Industrial Biotechnology processes.

References

[1] R. Goebel, R. G. Sanfelice, A. R. Teel, Hybrid dynamical systems, IEEE Control Systems 29 (2) (2009) 28–93.

[2] A. van der Schaft, J. Schumacher, An Introduction to Hybrid Dynamical Systems, Lecture Notes in Control and Information Sciences, Springer, 2000.

[3] Handbook of Hybrid Systems Control: Theory, Tools, Applications, Cambridge University Press, 2009.

[4] P. J. Antsaklis, A brief introduction to the theory and applications of hybrid systems (2000).

[5] R. Johansson, A. Rantzer (Eds.), Nonlinear and Hybrid Systems in Automotive Control, Springer-Verlag, Berlin, Heidelberg, 2002.

[6] K. Aihara, H. Suzuki, Theory of hybrid dynamical systems and its applications to biological and medical systems, Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences 368 (1930) (2010) 4893–4914.

[7] L. Bortolussi, A. Policriti, Hybrid systems and biology: Continuous and discrete modeling for systems biology, in: Proceedings of the Formal Methods for the Design of Computer, Communication, and Software Systems. 8th International Conference on Formal Methods for Computational Systems Biology, SFM'08, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 424–448.

[8] N. Lynch, R. Segala, F. Vaandrager, Hybrid i/o automata, Information and Computation 185 (1) (2003) 105 – 157.

[9] T. A. Henzinger, The theory of hybrid automata, in: Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science, LICS '96, IEEE Computer Society, Washington, DC, USA, 1996, pp. 278–.

[10] A. Bemporad, M. Morari, Control of systems integrating logic, dynamics, and constraints, Automatica 35 (3) (1999) 407–427.

[11] R. David, H. Alla, Discrete, Continuous and Hybrid Petri Nets, Springer, Berlin, 2004, (2^{nd} edition, 2010).

[12] M. Dotoli, M. Fanti, A. Giua, C. Seatzu, First-order hybrid Petri nets. An application to distributed manufacturing systems, Nonlinear Analysis: Hybrid Systems 2 (2) (2008) 408 – 430.

[13] T. A. Henzinger, P.-H. Ho, H. Wong-Toi, Hytech: A model checker for hybrid systems, in: O. Grumberg (Ed.), Computer Aided Verification, Springer Berlin Heidelberg, Berlin, Heidelberg, 1997, pp. 460–463.

[14] A. Bemporad, G. Ferrari-Trecate, M. Morari, Observability and controllability of piecewise affine and hybrid systems, IEEE transactions on automatic control 45 (10) (2000) 1864–1876.

[15] B. Kouvaritakis, M. Cannon, Model Predictive Control. Classical, Robust and Stochastic, Springer International Publishing, 2016.

[16] T. Murata, Petri Nets: Properties, Analysis and Applications, Procs. of the IEEE 77 (4) (1989) 541–580.

[17] M. Silva, Introducing Petri Nets, Practice of Petri Nets in Manufacturing (Chapman & Hall, London, 1993) 1–62.

[18] J. Júlvez, D. Dikicioglu, S. G. Oliver, Handling variability and incompleteness of biological data by flexible nets: a case study for Wilson disease, npj Systems Biology and Applications 4 (1) (2018) 7.

[19] F. Balduzzi, A. Giua, G. Menga, First-order hybrid Petri nets: a model for optimization and control, IEEE Transactions on Robotics and Automation 16 (4) (2000) 382–399. doi:10.1109/70.864231.

[20] A. D. Febbraro, D. Giglio, N. Sacco, Urban traffic control structure based on hybrid Petri nets, IEEE Transactions on Intelligent Transportation Systems 5 (4) (2004) 224–237. doi:10.1109/TITS.2004.838180.

[21] G. Cavone, M. Dotoli, C. Seatzu, Management of Intermodal Freight Terminals by First-Order Hybrid Petri Nets, IEEE Robotics and Automation Letters 1 (1) (2016) 2–9. doi:10.1109/LRA.2015.2502905.

[22] W. E. Hart, C. Laird, J.-P. Watson, D. L. Woodruff, Pyomo–Optimization Modeling in Python, Vol. 67, Springer Science & Business Media, 2012.

[23] I. Gurobi Optimization, Gurobi optimizer reference manual (2015).

[24] IBM ILOG CPLEX Optimizer (2010).

[25] M. Silva, J. Júlvez, C. Mahulea, C. R. Vázquez, On fluidization of discrete event models: observation and control of continuous Petri nets, Discrete Event Dynamic Systems 21 (4) (2011) 427.